

BOOKI DEV GUIDE

Copyright : The Contributors (see back)

Published : 2009-12-15

License : None

Note : We offer no warranty if you follow this manual and something goes wrong. So be careful!

TABLE OF CONTENTS

TABLE OF CONTENTS

UNKNOWN 0

1 Introduction	2
2 Introduction	5
3 Overview	6
4 Vision	7
5 Use Cases	9
6 Features	11
7 How It Works in Practice	12
8 How it Works	13
9 Creating an Account	14
10 Creating a Book	15
11 Book Settings	16
12 Editing a Book	17
13 Editing a Book	18
14 Chatting	19
15 Editing a Book	21
16 How We Work	23
17 How We Work	24
18 Technology Used in Booki	25
19 Core Components	31
20 Booki-Zip	32
21 How You Can Help	33
22 Getting Stuck In	34
23 Testing	35
24 Writing Docs	36
25 RSS	37
26 Visualisations	38
27 Groups	39
28 Book Admin	40
29 Export Formats	41
30 Import Formats	42

31 CSS Templates	43
32 Messaging	44
33 Copyright Tracking	45

2.1. INTRODUCTION

Booki is a Collaborative Authoring and Book Sprint platform, designed for the collaborative and rapid development of highly structured content (books).

With the exception of Booki, there is no software available for this practice. Some have tried wiki and CMS software and find its not easy. Wikis are for the asynchronous development of unstructured web content. CMS are for the management of web-structured online content. Booki is something new.

ABOUT THIS BOOK

This book is intended to give potential booki developers a very brief understanding of the big picture, the team, the internal mechanics, and the jobs that we need help with.

You are free to edit this book in booki at :
<http://www.booki.cc>

UNKNOWN 0

1. INTRODUCTION
2. INTRODUCTION
3. OVERVIEW
4. VISION
5. USE CASES
6. FEATURES
7. HOW IT WORKS IN PRACTICE
8. HOW IT WORKS
9. CREATING AN ACCOUNT
10. CREATING A BOOK
11. BOOK SETTINGS
12. EDITING A BOOK
13. EDITING A BOOK
14. CHATTING
15. EDITING A BOOK
16. HOW WE WORK
17. HOW WE WORK
18. TECHNOLOGY USED IN BOOKI
19. CORE COMPONENTS
20. BOOKI-ZIP
21. HOW YOU CAN HELP
22. GETTING STUCK IN
23. TESTING
24. WRITING DOCS
25. RSS
26. VISUALISATIONS
27. GROUPS
28. BOOK ADMIN
29. EXPORT FORMATS
30. IMPORT FORMATS

- 31. CSS TEMPLATES
- 32. MESSAGING
- 33. COPYRIGHT TRACKING
- 34. GIT
- 35. PDF LAYOUT
- 36. APPENDICES
- 37. BOOK SPRINTS

INTRODUCTION

Booki is a Collaborative Authoring and Book Sprint platform, designed for the collaborative and rapid development of highly structured content (books).

With the exception of Booki, there is no software available for this practice. Some have tried wiki and CMS software and find its not easy. Wikis are for the asynchronous development of unstructured web content. CMS are for the management of web-structured online content. Booki is something new.

ABOUT THIS BOOK

This book is intended to give potential booki developers a very brief understanding of the big picture, the team, the internal mechanics, and the jobs that we need help with.

You are free to edit this book in booki at :
<http://www.booki.cc>

4. VISION

Booki facilitates the collaborative production of books through simple and effective writing and book management tools. The Booki interface is designed around the authors and their needs to write, to discuss their views, to seek assistance with partner writers, to translate and reuse content. Using a platform like this you can collaborate to produce very high quality content, very quickly, or you can take your time and write a book by yourself.

However, rapid development of content is the core focus of Booki. Through FLOSS Manuals we have experimented with the rapid development of content made possible by Book Sprints. Book Sprints are intensive collaborative events where 6-20 people focus on writing a book in 2-5 days. These collaborators might be in the same room or located around the net, in either case they are networked.

Booki is bent towards the accelerated production models enabled by Book Sprints. Hence the feature set is focused on matching the rapid pace of publishing possible in the modern era of print on demand with a rapid pace of content creation. In Booki print-ready source (book formatted PDF) is generated on the server in a matter of minutes. This means you can finish a book in 2-5 days, and then have the print ready source available in another 2-5 minutes. Edit the book and then generate another PDF with your changes immediately.

To work in this context means ditching Desktop Publishing models and replacing the design of books in the hands of web technologies. DTP is too slow for these processes. In Booki the design of book formatted PDF is controlled with CSS – bringing book design to the web literate. If you know CSS you can change the style your book.

Additionally, Booki posits the idea that books can always be improved. Hence books should always be reusable. Booki uses open content licenses by default which partially assists reuse. However without a repository of content and a technical process for managing reuse, content won't often be reused regardless of the license. Booki makes reuse easy by technically facilitating the forking, updating, improving, remixing, rewriting, recontextualisation, and translation of books.

Translation is an especially interesting case of reuse. Where the traditional publishing industry cannot easily translate content due to its restrictive licensing and contractual requirements, Booki encourages and facilitates translation wherever possible. Booki provides translation tools to ease the migration of books across language boundaries. However translation is not the only language feature of Booki. Booki also places a high value on the production of original content in any language. For this reason Booki is designed to be easily localised.

Actually Booki makes bold challenges to what a book is. Booki is really a platform for the collaborative development of Comprehensive Texts, one output of which is books. However Booki can output your text to multiple formats, opening the door to RSS syndication, ebook distribution, or any other format for the distribution on offline or online media. Booki helps comprehensive texts move fluidly through different media. However, 'comprehensive text' is too wordy and so we still use the term 'book', but perhaps more liberally than traditional publishers.

Although built to support Collaborative Authoring and Book Sprints you can use Booki to write a book by yourself over a longer period of time. You can also install your own version of Booki for your own use, or share an existing installation. Whichever strategy you take; collaborating, writing in solo, rapidly developing books, or taking your time - when you have worked with Booki you will never think of publishing the same way again.

5. USE CASES

Booki supports any individual or organisation that wishes to write a book online and output this material into numerous formats including text, HTML, and PDF. The environment is content agnostic. Some typical uses of the platform might be :

- Writing printed books eg. The Inkscape Manual, a cook book
- Writing any content as an individual
- Collaboratively authoring content – e.g. the vast bulk of FLOSS Manuals material
- Using the platform during Book Sprints
- Remixing content for specific uses. For example – creating workshop manuals
- Customising existing content to apply to a very specific context. For example, a programmer might use the FLOSS Manuals content and be able to customise it to a specific client
- Embedding information in my web page so it looks like my webpage
- Translating a book into another language

Simple use cases follow.

Book Sprint : I am in an advocacy group working for rights of immigrant workers. We need to create a booklet about workers rights. I read the (free) FLOSS Manuals online manual on Book Sprints and then organise 5 staff to dedicate 3 consecutive days the following week. The first morning we each create an account. I have already created the new book, given it an open license, sent the others the URL, and prepared the Index. We spend 3 hours discussing the chapters and decide who will write what. At lunch on the first day we start writing. We continue to write for the following 2 days. At the end of the last day we output an A5 book formatted PDF of 110 pages and take it to our local printer to make 500 stapled copies. Over the next 2 days peers from other offices log in to improve and build upon our work. We make the book formatted PDF freely available from our website to read or to print for distribution. Each of the other offices print copies locally and distributes them.

Re-contextualising Content : I am a developer that has contributed to several manuals on Free Software. My specialty is Drupal, and I have a client that I have just created a site for using Drupal. I want to create a manual for that client so I go to my user page and fork the existing FLOSS Manuals Groups manual on Drupal. I now have a standalone version of the manual on my account, any alterations I do will not be reflected in the original manual unless I merge the changes up stream. I remove several chapters not relevant to the client, add 4 new chapters that are specific to the client. I mark the new chapters as “Copyright : all rights reserved” and private so they are not reusable or seen by anyone else using the system. I write these chapters with information that is specific to my client and replace several existing screen shots in other chapters with screenshots of my clients site. I export the material to book formatted PDF, upload it to lulu.com print on demand service and buy 2 copies of the perfect bound book for my client. Later a potential client calls me for a meeting - I use my version of the Drupal manual, remove the client sensitive material, upload it, buy it, and bring the book to the meeting.

Translating : I am a volunteer translator. I want to translate a book on Mexican Cooking from the CookMe Group into French. I fork the existing manual from the CookMe Group to my user page. I force a machine translation of the content and mark all chapters “to be translated”. I message others subscribed to the Booki French group that I need some help and with 3 others we translate the material and mark all the chapters “to be proofed” or “requires technical proofing” as we complete them. When we are finished I invite a friend to proof the content for us. She takes 1 week, doing it in her spare time. After each chapter is proofed she marks them “complete”. Later I output the translated book to book formatted PDF, upload to lulu.com and buy a copy.

Multiple Formats : I am an educator and have written a text book on digital design using Booki. While writing I kept the book private but invited select users to view the material and leave comments – this was great feedback for me. Now the book is completed and has been picked up by a publisher – I output an .odt file from Booki and open this in OpenOffice.org, save to a .doc format (the publisher does not know

about free software) and send to them. I also negotiated a Creative Commons license for the material with the publisher. I release the book on my website using the embed api and invite translations. Within 4 weeks there is a completed Spanish translation which is uploaded to a print on demand service and sold through my website. A Portuguese translation is ready 3 weeks later. After an enthusiastic response to the Portuguese translation in Brazil I output the material to multiple column PDF suitable for printing on large format newsprint. A school district in Sao Paolo finds a local printer, downloads the PDF for free, and produces 1500 copies on newsprint and distributes for free around schools – this costs the same as printing or buying 30 printed books so I feel good and the schools make a huge savings.

Portability : I am invited to facilitate a Book Sprint on farming techniques in a remote area. The venue is a farm with no internet connectivity. I know a little about how to install web services so I download and install Booki on my laptop. At the Book Sprint we have 4 laptops connected to the local network version of Booki running on my machine. We write the book in 5 days. We output the book formatted PDF from my laptop, print the book at a local printer and distribute locally as educational materials. Later I upload the sources to the version of Booki running on my own server so others can access and reuse the content as they please.

6. FEATURES

To understand Booki in more detail let's look at its principle features. The following is a brief description of the main features and some information on the progress so far. The list is not exhaustive and we invite brainstorming on features at anytime (although we want to remain focused on achievable milestones).

Versioning

not currently implemented

One of the most powerful features of Booki is the ability to easily copy and reuse content. Booki will use GIT as the versioning repository for the content. This means we can easily fork books into new versions for reusing, rewriting, translation, or recontextualising the content. Changes can be merged upstream if desired.

Book Formatted PDF Output

implemented (Objavi) -- <http://objavi.flossmanuals.net>

The server side creation of Book Formatted PDF is a pivotal feature of Booki. Booki creates book formatted PDF using the Webkit PDF rendering engine. With Booki if you want to update a book, you edit it in the browser and create another print ready PDF in a matter of minutes. Furthermore, the book formatted PDF supports Unicode, bi-directional text, and reverse binding for printing right-to-left texts on a left-to-right press and vice versa. The formatting engine will output customisable sizes including split column PDF suitable for printing on large scale newspaper.

CSS Book Design

implemented- <http://objavi.flossmanuals.net>

The PDF rendering engine has CSS support. This changes the language of design from Indesign to CSS which means any web native can control the design of the book.

Outputs

partially implemented - <http://objavi.flossmanuals.net>

Booki is really a Comprehensive Text authoring environment, one output of which is books. Users also will be able to publish to static HTML for linking from a website, to .odt (OpenOffice rich text format), standalone HTML (for downloading as a zip/tar and including in offline media - also not yet implemented), and screen readable PDF. Other XML, and 'e-book' output options can be developed as Extensions.

Real Time Collaboration Tools

partially implemented

Book Sprints, remote, and asynchronous authoring all have their own communication requirements. Harmonising these into one system is where FLOSS Manuals has done a lot of real world research and Booki will take the best of what we have learned. Booki will include tools such as real time edit notifications to see who is editing what at this moment, a real time chat (web / IRC gateway), system level notifications, chapter status markers, book progress indicators, work flow tools, and live user status listings.

Localisation

not currently implemented

Booki needs to be available in any language where it is needed. Hence we will integrate the mature Pootle code base into Booki to enable localisation of the environment.

Translation

not currently implemented

Content can be forked and marked for translation. A translation version of a book will provide link backs to the original material, be placed in a translation work flow, and edited in a side-by-side view where the translator can also see the original source. We are considering a World Wide Lexicon Extension to add machine translation to the work flow

Importing

implemented - <http://objavi.flossmanuals.net/espri.cgi>

Booki is also useful for importing book content. Currently Booki supports the import of Archive.org ebooks (1.6 million), Wikibooks (from Wikibooks.org), and any epub online.

Installable

not currently implemented

Booki is installable to run as a web service or locally. Installation can be done from source that will require

python knowledge. Ubuntu users will be able to easily install Booki using the .deb packages (no knowledge of python required).

Booki Service

implemented - <http://www.booki.cc>

FLOSS Manuals will itself use Booki for development of manuals about Free Software. We will also run a free Booki service on the domain <http://www.booki.cc> for those who do not wish to install their own version.

Embed API

not currently implemented

Many users will want to use Booki to author material to host under their own domain. Booki will provide an API similar to the FLOSS Manuals beta API for embedding books with indexes in their own websites. This means users need only cut and paste a few lines of HTML into their website and the book will appear as if hosted under their domain.

User Pages

not currently implemented

Booki will put the user in the center of the design. From the users point of view this makes Booki user-centric, not content centric. Your user page is the homepage when you log in and from here you can monitor changes in books, communicate with others, and participate. From this page you can create books, create and join groups, make announcements about Book Sprints or other activities, track changes on your own or others books, and subscribe to user services.

Dynamic Groups

not currently implemented

Booki groups are self associating collections of users that collaborate to write books on a given topic. For example, FLOSS Manuals would be one such group, which aggregates manuals about Free Software. If I want to help others write Free Software manuals I would subscribe to the FLOSS Manuals group.

RSS

not currently implemented

Books can be syndicated with RSS.

Copyright Tracking and Management

not currently implemented

All attributions are automated in Booki. We also desire to reduce the overhead for finding the rights holder for any work authored in Booki so the process of obtaining re-license and re-use permission will be partially automated. Meaning that the technical process of managing licenses is taken away from the user, making the permissible reuse of content very easy.

Book Management

implemented

In Booki new chapters can easily be added to the Index, redundant ones removed, and the Index can be managed with a drag and drop interface.

License Control

not currently implemented

Booki will enable the user to decide the license for any book they create. Booki will ensure, as far as it is technically feasible (its difficult to prevent simple cut and paste copyright violations), that content from within a Booki content base is not recombined with incompatibly licensed material. Licenses can also be managed on a chapter level. Open content licenses will be used by default.

8. HOW IT WORKS

The following chapters are an insight into how Booki actually works now. You can see most of the functionality online at <http://www.booki.cc>

If you do create an account on the above URL please be aware that at the time of writing we have no back-up processes in place and it is alpha software. We offer no apologies if content gets lost.

The features that are not implemented are largely kept in the 'How you can help' section unless they relate to functionality that already exists.

9. CREATING AN ACCOUNT

Account generation is simple - add your desired username, email, and password to the front page.

□

Booki will generate the account and log you in.

This is implemented and working at present.

FEATURES TO BE ADDED

We wish to implement OpenID. Also, once logged in the user should be asked for a few extra details about themselves :

- * photo
- * brief description of themselves
- * bio
- * contact details

10. CREATING A BOOK

Once a user has created an account they can manage the creation of books from their user page. This is as simple as entering the title of the book in the field provided and choosing the license from the drop down menu :

□

The book will then be generated and listed in the list of book on their user page (no refresh needed). The book has by default no chapters, sections, or content.

This is currently implemented.

FEATURES TO BE ADDED

The user should be able to add more information about the book once it has been created. Currently no meta data is able to be entered such as description, copyright notices, direction of the text (right-to-left/left-to-right), dual licensing etc. We want to keep the book generation process down to the basics so extra information and settings for the book can be managed in the tabs "Settings" and "Info" once the book has been generated.

11. BOOK SETTINGS

12. EDITING A BOOK

13. EDITING A BOOK

14. CHATTING

15. EDITING A BOOK

16. HOW WE WORK

We are happy to extend our core team in time to include others that are enthusiastic about the project and can commit the time and energy. Currently we are a small team consisting of just the following members :

Adam Hyde - Project Manager

Aleksandar Eraković - Programmer (Booki)

Douglas Bagnall - Programmer (Espri, Objavi)

There are several others involved such as Jan Gerber and Franco Iacomella. We also work closely with the Archive.org team on several issues including epub import and export of content directly to the Archive.org servers.

The development of the FLOSS Manuals platform was really a process of experiencing first hand what is needed to develop Free Software manuals through two processes :

1. Community and individual writing of documentation
2. Book Sprints

With FLOSS Manuals the team would work with the community and Book Sprint teams creating documentation and then discuss what works and doesn't work and extend the platform accordingly. Sometimes this was done with the assistance of project funding, and sometimes without. In all cases we preferred (and still do) utility over complicated design. KISS.

We have now ceased work on the FLOSS Manuals platform to develop Booki and I think it's fair to say that while the FM platform is fully functional and productive, we see it not as a 'finished' work but as a prototype and our primary feature-set design document for the current Booki development.

This short pre-history also explains our preferred development methodology. Although we do have milestones and we have a pretty clear idea of what we want to build, much of the detail and new ideas are worked out in discussion in IRC. If you wish to contribute to the code base then the process of getting involved is more about engaging with the resources listed below and asking questions than it is about reading our (non-existing) design documentation.

If you wish to help fix existing features please look at the Trac tickets.

You can also read the section 'How You Can Help' if you wish to tackle implementing new features.

The following are the main channels for communication and for getting the code.

RESOURCES

IRC : [irc.freenode.net #fm-tech](irc://irc.freenode.net/#fm-tech)

Email : <http://lists.flossmanuals.net/listinfo.cgi/booki-dev-flossmanuals.net>

Code

<http://booki-dev.flossmanuals.net/git>

Here you will find the repositories for Booki, Objavi and Espri

Trac

<http://booki-dev.flossmanuals.net/>

Primary Install

<http://www.booki.cc>

FLOSS Manuals

17. HOW WE WORK

We are happy to extend our core team in time to include others that are enthusiastic about the project and can commit the time and energy. Currently we are a small team consisting of just the following members :

Adam Hyde - Project Manager

Aleksandar Eraković - Programmer (Booki)

Douglas Bagnall - Programmer (Espri, Objavi)

There are several others involved such as Jan Gerber and Franco Iacomella. We also work closely with the Archive.org team on several issues including epub import and export of content directly to the Archive.org servers.

The development of the FLOSS Manuals platform was really a process of experiencing first hand what is needed to develop Free Software manuals through two processes :

1. Community and individual writing of documentation
2. Book Sprints

With FLOSS Manuals the team would work with the community and Book Sprint teams creating documentation and then discuss what works and doesn't work and extend the platform accordingly. Sometimes this was done with the assistance of project funding, and sometimes without. In all cases we preferred (and still do) utility over complicated design. KISS.

We have now ceased work on the FLOSS Manuals platform to develop Booki and I think it's fair to say that while the FM platform is fully functional and productive, we see it not as a 'finished' work but as a prototype and our primary feature-set design document for the current Booki development.

This short pre-history also explains our preferred development methodology. Although we do have milestones and we have a pretty clear idea of what we want to build, much of the detail and new ideas are worked out in discussion in IRC. If you wish to contribute to the code base then the process of getting involved is more about engaging with the resources listed below and asking questions than it is about reading our (non-existing) design documentation.

If you wish to help fix existing features please look at the Trac tickets.

You can also read the section 'How You Can Help' if you wish to tackle implementing new features.

The following are the main channels for communication and for getting the code.

RESOURCES

IRC : [#fm-tech](irc://irc.freenode.net)

Email : <http://lists.flossmanuals.net/listinfo.cgi/booki-dev-flossmanuals.net>

Code

<http://booki-dev.flossmanuals.net/git>

Here you will find the repositories for Booki, Objavi and Espri

Trac

<http://booki-dev.flossmanuals.net/>

Primary Install

<http://www.booki.cc>

FLOSS Manuals

18. TECHNOLOGY USED IN BOOKI

Booki is Python based, built with the (bare) Django framework, and using JQuery for dynamic page elements. Postgres is the main database, and we use a Redis database for caching.

XHTML is the file format for content. Booki uses the Xinha WYSIWYG editor for creating content. epub is increasingly becoming the default import format for content coming from external projects.

The PDF generation requires Webkit as a rendering engine - we have chosen this above other PDF rendering engines since Webkit has unicode, CSS, and bi-directional text support. However we would rather use the Mozilla Firefox layout engine (NGLayout & XPFE projects but usually known somewhat incorrectly as Gecko) and this will be implemented as the default renderer in time.

There are other tools required for some outputs from Objavi, for example the rendering of .odt requires OpenOffice to be installed with unoconv.

The format for exchanging books between the three core components (Espri, Objavi, and Booki) is a format defined by the project called Booki-zip. Booki-zip contains mostly xhtml files and a JSON file.

The Web/IRC gateway optionally require a standalone IRC service running locally.

For development we will use Apache2 for http delivery, but other deployments can use any http service.

TO BE IMPLEMENTED

Content will eventually be stored in GIT. Pootle will also eventually be integrated into the core of Booki, and hence interface texts will be kept in Portable Object files (.po). Rabbitmq will be implemented to manage message queues (such as import or export requests). We would like to implement OpenID for logins.

LICENSE

Each release will be as source. Beta and later releases will also be available as Debian .deb packages. The license is GPL2+.

RESOURCES

Django - <http://www.djangoproject.com/>

JQuery - <http://jquery.com/>

Postgres - <http://www.postgresql.de/>

Redis - <http://code.google.com/p/redis/>

Xinha - <http://trac.xinha.org/>

epub - <http://www.idpf.org/specs.htm>

Webkit - <http://webkit.org/>

Mozilla Layout Engine - <http://www.mozilla.org/newlayout/>

GIT - <http://git-scm.com/>

Pootle - <http://translate.sourceforge.net/wiki/pootle/index>

OpenID - <http://openid.net/>

Rabbitmq - <http://www.rabbitmq.com/>

19. CORE COMPONENTS

Booki is made from three separate projects - Booki, Objavi, and Espri.

BOOKI

Booki is the core component and what most users refer to as 'Booki'. This component is largely developed by Aco and it looks after user and group management and the creation and editing of books. Booki also houses all the communication tools required for collaborative authoring.

When a user requests a book to be imported Booki sends a request to Espri, Espri then sends Booki a Booki-zip file.

When a user publishes a book Booki sends Booki-zip files to Objavi for creating outputting formatted PDF etc.

OBJAVI

Objavi is the output ('publishing') engine. Objavi 1 was developed by Luka Frelih in 2008, Objavi 2 in 2009 by Douglas Bagnall. Objavi takes Booki-zip files from Booki and currently outputs to PDF (web readable, book formatted, and newsprint formatted), ODT, and epub.

Objavi is primarily used for generating book formatted PDF which is uploaded to a print on demand service. The PDF layout is controlled by CSS, supports bi-directional text and is unicode compliant.

Objavi can be accessed directly as a web service or accessed via the Publishing tab in the edit interface of any book.

ESPRI

Espri is the importer - handling the import of books from the internet and converting them to Booki-zip. It was developed in 2009 by Douglas to import manuals from FLOSS Manuals but it now has assumed the role of a general importer. Espri prefers to receive either Booki-zip files directly, or epub. If neither of these are available then Espri manages the conversion of the content into Booki-zip through external conversion scripts. The general case for Espri works as follows :

□

Espri currently does the following :

1. **Archive.org Import** : import directly from Archive.org. In this case the user inputs the Archive.org ID for a book into the appropriate field on their user page. Booki then sends this request to Espri. Espri will request an epub for the book directly from Archive.org. Espri then converts this epub into Book-zip and sends it back to Booki.
2. **Wikibooks Import** : imports books from wikibooks.org. In this case the user must input a Wikibooks URL into the appropriate field on their user page. Booki then sends this request to Espri. Espri then launches Wiki2epub which in turn retrieves the files from wikibooks.org, creates an epub and sends this file to Espri. Espri then creates a Booki-zip file and sends this to Booki.
3. **Import Epub** : imports any epub online. The user enters a URL for an epub into the appropriate field on their homepage. Booki then sends this request to Espri. Espri retrieves the epub, creates a Booki-zip and hands this back to Booki.

20. BOOKI-ZIP

The Booki-zip format is the archive of files representing a book that is sent between the core components. The format Booki-zip 1 has been finalised and is documented here.

BOOKI-ZIP 1

This describes the booki-zip format that Booki, Espri, and Objavi use to communicate with each other.

Zip container format

A booki-zip file is a zip file[1], with certain restrictions. The ultimate test of whether a zip is correctly encoded is whether its contents can be extracted by the zipfile modules in Python 2.5 and 2.6. This means the contents must be either uncompressed or deflate-compressed. ZIP64 extensions are OK (though unnecessary in practical terms), but encryption and comments are not.

The first file in the zip should be uncompressed and named "mimetype". It should contain only the 23 characters "application/x-booki+zip". This string will end up in the first few bytes of the zip file, allowing it to be identified without unzipping.

Directory structure

As well as the just mentioned "mimetype", the booki-zip must have a file called "info.json" in its root directory, the contents of which will be described shortly. Any other files in the root directory should be html files intended for editing with Booki. Any associated files that are not directly editable by Booki should be in a subdirectory named 'static'. Here is an example structure:

```
/
mimetype
Introduction.html
UseCases.html
AdamsTips.html
Credits.html
info.json
static/
  BookSprints-ott-adam-en.jpg
  Blog-writers-en.png
  Floss-100-en.gif
  example.css
```

All references from the html to the files in 'static' should use relative addresses. For example, an image should be linked thus:

```

```

It is recommended but not required that the file names have conventional extensions (".html", ".jpg", etc). File names should not contain spaces, and must meet the restrictions imposed by the zip format.

There should be nothing in the root directory other than "mimetype", "info.json", and the html files, and there should be no other subdirectories other than "static". Apart from starting with "mimetype", there is no required order to the arrangement of entries within the zip file itself. Other than "mimetype", files should be deflated-compressed.

character encoding

All html files, and info.json, should be encoded as utf-8.

info.json

The "info.json" file describes the structure of the document and carries metadata. It is a JSON file [3], containing a single JSON object with 5 members, as shown here:

```
{
  "version": 1,
  "spine": [ ... ],
  "TOC": [ ... ],
  "manifest": { ... },
  "metadata": { ... }
}
```

Being JSON object members, the ordering of these elements is not significant. The following order is for narrative purposes only.

info.json version

This indicates which version of the booki-zip standard is being used. This document describes version 1. If the version is not 1, nothing else here necessarily applies.

info.json manifest

The manifest is a mapping of identifiers to file names and mime-types. Each entry looks like:

```
identifier: {
  "filename": filename,
  "mimetype": mimetype,
  "contributors": contributors,
  "rightsholders": rightsHolders,
  "license": license
}
```

The constraints on **identifier** match the XML name specification[4] (in short, avoid spaces and most punctuation). In practise, the **identifier** is often related to the **filename**.

filename locates the file within the zip, and must match a path in the zip index.

mimetype is the IANA media type [5] of the file. Booki-editable html files must be of type 'text/html', and other files should be correctly identified.

contributors is a list of names of people who have contributed to this file. It can be empty.

rightsHolders is a list of the people or organisation that manages the rights for the chapter

license - a list of licenses applicable to the chapter. If more than one license is listed, the disjunction of these licenses applies. For the common licenses which have abbreviations listed in the license section below, the abbreviation should be used. Other licenses should be listed as an url, which could be a relative url to a file in the booki-zip. Copyrighted files with no sharing license should have an empty list ("[]"), and files out of copyright should have "public domain" as their single member.

The manifest shouldn't list the 'mimetype' or 'info.json' files, just the editable html and associated static files.

An example manifest, containing two html files and an image, is shown here:

```

"manifest": {
  "Introduction": [
    "url":      "Introduction.html",
    "mimetype": "text/html",
    "contributors": ["Adam Hyde", "Aleksander Erkalovic"],
    "rightsholders": ["Adam Hyde"],
    "license":    ["CC-BY-SA"]
  ],
  "arbitrary-identifier_0005": [
    "url":      "UseCases.html",
    "mimetype": "text/html",
    "contributors": [],
    "rightsholders": ["Wikimedia Foundation"],
    "license":    ["FDL", "CC-BY-SA"],
  ],
  "BookSprints-ott-adam-en.jpg": [
    "url":      "static/BookSprints-ott-adam-en.jpg",
    "mimetype": "image/jpeg",
    "contributors": ["Ansell Adams"],
    "rightsholders": ["Ansell Adams"],
    "license":    []
  ]
}

```

info.json spine

The spine lists the identifiers of all the html files in the order they appear in the book. It looks like:

```
"spine": [ identifier, identifier,... ]
```

where each **identifier** is the manifest identifier for an editable html page.

Here is a possible spine for the manifest used in the previous example:

```
"spine": ["Introduction", "arbitrary-identifier_0005"]
```

info.json TOC

The TOC (Table of Contents) specifies navigation points with the book. It uses a nested structure, with less significant divisions being contained within the "children" attribute of the greater division.

The "TOC" element itself is a list of objects with the following structure:

```

{
  "title":  division title (optional),
  "url":    filename and possible fragment ID,
  "type":   string indicating division type (optional),
  "role":   epub guide type (optional),
  "children": list of TOC structures (optional)
}

```

title is a free string giving the divisions title. It may be omitted.

url points to the start of the division. It should consist of a filename as found in the manifest, optionally followed by a '#' and a fragment identifier.

type is a string indicating what kind of navigation point it is. This might be used to determine text styles.

role, if present, indicates the navigation point has a particular structural role. It must be a keyword for "reference type" as defined in the guide section of the epub OPF specification[6].

children, if present, contains a list of objects following this same specification. These are subsections of this section.

An example:

```
"TOC": [
  {
    "title": "INTRODUCTION",
    "url": "Introduction.html",
    "type": "booki-section",
    "children": [
      {
        "title": "WHAT IS GSoC?",
        "url": "Introduction.html",
        "type": "chapter",
        "role": "text"
      },
      {
        "title": "WHY GSOC MATTERS",
        "url": "Testimonials.html",
        "type": "chapter",
        "children" [ ... ]
      }
    ]
  }
]
```

info.json metadata

The names in the metadata object are "namespaces" in which "keywords" are defined. The objects referred to by keywords are further divided by "scheme". Each scheme points to a list of values. If the keyword is indivisible, there should be a single scheme identified by an empty string (""). Further, if a scheme is the primary default for that keyword, it may be identified by an empty string as well as by its scheme name.

Here's the diagram:

```
"metadata": {
  namespace: {
    keyword: {
      scheme: [value, value,...],
      scheme: [value],...
    },...
  },...
}
```

Booki uses Dublin Core[7] metadata keywords wherever possible, which are stored under the namespace "http://purl.org/dc/elements/1.1/".

An example metadata section is shown below:

```
"metadata": {
  "http://purl.org/dc/elements/1.1/": {
    "publisher": {
      "": ["FLOSS Manuals http://flossmanuals.net"]
    },
    "language": {
      "": ["en"]
    },
    "creator": {
      "": ["The Contributors"]
    },
    "contributor": {
```

```

    "": ["Jennifer Redman", "Bart Massey", "Alexander Pico",
        "slena deckelmann", "Anne Gentle", "adam hyde", "Olly Betts",
        "Jonathan Leto", "Google Inc And The Contributors",
        "Leslie Hawthorn"]
  },
  "title": {
    "": ["GSoC Mentoring"]
  },
  "date": {
    "start": ["2009-10-23"],
    "last-modified": ["2009-10-30"]
  },
  "identifier": {
    "flossmanuals.net": ["http://en.flossmanuals.net/epub/GSoCMentoring/2009.10.23-19.49.01"],
    "archive.org": ["gsocmentoring00fm"]
  },
  "rights": {
    "": ["Copyright The Contributors. Licensed under the GPLv2. See Appendix.html in this zip file or
http://www.gnu.org/licenses/gpl-2.0.txt for details"]
  },
  "http://booki.cc/": {
    "server": {
      "": ["en.flossmanuals.net"]
    },
    "book": {
      "": ["GSoCMentoring"]
    },
    "dir": {
      "": ["LTR"]
    },
    "license": {
      "": ["GPL"]
    }
  }
}

```

There must be "language", "creator", "identifier", and "title" Dublin Core elements present. The "contributor" Dublin Core element should list all contributors to individual files, as listed in the manifest, unless those contributors are already identified in the "creator" field.

The Dublin Core "rights" element should contain a human readable summary of the book's copyright and license.

The "http://booki.cc/" namespace can contain the following elements:

server the Booki or FLOSS Manuals server on which the book is edited.

book the book's identifier on that server.

dir the primary text direction ('LTR' or 'RTL'). If unspecified, 'LTR' is assumed, though software may determine the text direction by inspecting the contents.

license licenses used in the book, using if possible the abbreviations in the next section. Multiple licenses listed here do not necessarily indicate a disjunction of these licenses applies to each file; rather it might mean each license applies to a different subset of the files. All licenses used should be included in the text of the book.

Other namespaces are permitted but will not be used by Booki. They will, as far as possible, be preserved through Booki edits and be exported to other formats.

license abbreviations

The following abbreviated license identifiers should be used for the licenses defined at the corresponding URLs.

GPL	http://www.gnu.org/licenses/gpl.txt
GPLv2	http://www.gnu.org/licenses/gpl-2.0.txt
GPLv2+	http://www.gnu.org/licenses/gpl-2.0.txt [or greater version]
GPLv3	http://www.gnu.org/licenses/gpl-3.0.txt
GPLv3+	http://www.gnu.org/licenses/gpl-3.0.txt [or greater version]
LGPL	http://www.gnu.org/licenses/lgpl.txt
LGPLv2.1	http://www.gnu.org/licenses/lgpl-2.1.txt
LGPLv2.1+	http://www.gnu.org/licenses/lgpl-2.1.txt [or greater version]
LGPLv3	http://www.gnu.org/licenses/lgpl-3.0.txt
BSD	http://www.debian.org/misc/bsd.license
MIT	http://www.opensource.org/licenses/mit-license.html
Artistic	http://dev.perl.org/licenses/artistic.html
CC-BY	http://creativecommons.org/licenses/by/3.0/
CC-BY-SA	http://creativecommons.org/licenses/by-sa/3.0/
public domain	[no copyright]

Licenses not shown here should be listed as a URL that points to their text. It is possible for the URL to point to a local file within the booki-zip, but it is preferable to use a stable external link if one exists.

REFERENCES

- [1] Zip specification: <http://www.pkware.com/documents/casestudies/APPNOTE.TXT>
- [2] zipfile module: <http://docs.python.org/library/zipfile.html>
- [3] JSON specification: <http://json.org/>
- [4] XML name specification <http://www.w3.org/TR/REC-xml/#NT-Name>
- [5] Media types <http://www.iana.org/assignments/media-types/>
- [6] Guides in epub http://www.idpf.org/2007/opf/OPF_2.0_final_spec.html#Section2.6
- [7] Dublin Core metadata elements <http://dublincore.org/documents/2004/12/20/dces/>

22. GETTING STUCK IN

In general we prefer to talk through features with developers before they are built. This is to minimise duplicated efforts and co-ordinate when there might be more than one person working on a feature 'offline'. However, if you don't like that idea just get stuck in - get a copy of the code using GIT and submit as many changes as you like!

The following chapters document features that we need to implement and where we would appreciate help. Some features are large, others are possibly just a bite-size afternoons workload. Major features where research and discussion are required are marked 'major'.

23. TESTING

Anyone can test Booki at the site : <http://www.booki.cc> and file bugs, you don't have to be a programmer to do this. Bugs are defects in the software that the programmers need to fix. The following information on how to report a bug is largely with thanks to Karl Fogel and the Producing Open Source book.

To report a bug first make sure it's a bug. If you're not sure ask on the FLOSS Manuals mailing list first (<http://lists.flossmanuals.net/listinfo.cgi/discuss-flossmanuals.net>), or ask in IRC, irc.freenode.net, channel #flossmanuals.

If it is a bug then visit the Booki Trac (issue tracker/bug reporter) website at <http://booki-dev.flossmanuals.net>

Create an account on that site and choose "New Ticket". Here you will see fields to describe the issue. The most important information to get right is the name of the bug (this should be short but descriptive and input into the Summary field) and the Description.

In the Description write a simple description and steps on how to reproduce it (if possible). The simpler the reproduction recipe, the more likely a developer is to successfully reproduce the bug and fix it.

When you write up the bug use cut-and-paste to supply any output of error messages.

In addition to the reproduction recipe attach screenshots if you necessary (these can be very helpful) and also please provide the following information:

- * your booki username
- * the book you were working on (provide the URL)
- * the Operating System you are using
- * the browser you are using (including the version number if you know it)

Thanks. We know it's a lot of work to file an effective bug report, but a good report can save hours of a developer's time, and make the bug much more likely to get fixed.

24. WRITING DOCS

There are plenty of docs still to be written. We do not currently have :

- * user documentation (how to use the system)
- * installation documentation (how to install all components - these should be plain text documents included with the source code)
- * Objavi API documentation (how external applications can interact with Objavi)
- * Booki API documentation (how external applications can interact with Booki)
- * design documentation (description of the architecture of the code)

Please feel free to start any of these! Its easy to start in Booki ;)

25. RSS

Users need to be able to offer RSS feeds on a per-book, per-group, or per-account (their own) basis for others to track activities.

26. VISUALISATIONS

Inspired by <http://benfry.com/traces/>

Booki has a visualisation format which will be accessed with a simple URL. The URL can give version information about a single chapter, an entire book, or multiple books. The visualisation format is both for the Booki development community to use for creating visualisations of how books are developed, and the format is also intended for use by artists interested in data visualisation.

```
<?xml version="1.0" encoding='UTF-8'?>
<booki-revision>
<manual>
  <chapter name= "" url="">
    <version number="">
      <contributor></contributor>
      <date>...</date>
      <time>...</time>
      <text>...</text>
    </version>
  </chapter>
</manual>
</booki-revision>
```

The xml would be called by an external application with specified URL parameters. The exact detail of the parameters has not been worked out but could be mapped onto the elements represented in the above schema.

27. GROUPS

28. BOOK ADMIN

29. EXPORT FORMATS

30. IMPORT FORMATS

31. CSS TEMPLATES

32. MESSAGING

33. COPYRIGHT TRACKING

34. GIT

35. PDF LAYOUT

Thanks for reading!

Visit <http://flossmanuals.net> to make corrections or to find more manuals.